Ninja Runner

Xavier Clemente Rodríguez

Resumen— Ninja runner es un proyecto que tiene como finalidad desarrollar una aplicación para dispositivos Android, siguiendo las fases de creación de videojuegos de principio a fin. Para su realización, se ha utilizado el entorno de programación Unity 4.0 en 2D, que cuenta con el lenguaje de programación C# para el scripting. Este videojuego tiene las bases de un infinity runner con temática ninja, cuya característica es que el personaje principal avanza por un escenario generado de forma aleatoria e infinita, hasta que el jugador quede eliminado. El usuario deberá evitar todos los obstáculos e intentar realizar la máxima puntuación saltando. La mayoría de estos obstáculos/enemigos cuentan con un componente de inteligencia artificial: La creación del escenario es aleatoria, la dificultad y las recompensas aumentan en función de los puntos obtenidos, los enemigos se mueven o atacan teniendo en cuenta la posición del jugador y el enemigo final actúa en consecuencia de cómo ha jugado el usuario durante el camino previo al enfrentamiento.

Palabras Clave— Viedeojuego, Inteligencia artificial, Unity, Plataformas, 2D, C#, Android, Infinity Runner

Abstract— Ninja runner is a project that aims to develop an application for Android devices, into following the phases of the videogame creation process from beginning to end. In regard to its creation, the programming environment that was used is Unity 4.0 in 2D, which is supported by the programming language C# for the scripting. This videogame has the bases of an infinity runner with a ninja theme, whose characteristic is that the main character advances in a setting that is generated randomly and endlessly, until the player is eliminated. The user has to avoid all the obstacles and try to reach the highest score by jumping. Most of these obstacles/enemies possess an artificial intelligence component: the creation of the setting is random, difficulty and rewards increase depending on the points obtained, the enemies make their moves or attack by taking account of the player's position and the final enemy acts as a result of how the user has played during the prior journey of the duel encounter.

Index Terms— Videogame, Artificial intelligence, Unity, Platforms, 2D, C#, Android, Infinity Runner

1 Introducción

Los videojuegos llevan siendo una fuente de entretenimiento muy importante desde hace décadas. Pese a que el target principal de las empresas productoras de videojuegos siempre ha sido la gente joven, cada vez es más común ver gente adulta, incluso familias enteras disfrutando de su entretenimiento.

Si nos ponemos a pensar detenidamente, es normal que esta fuente de ocio esté en auge. Debido a la sociedad capitalista en la que nos encontramos, donde principalmente cada individuo está sometido a una rutina, jugar con una máquina acaba siendo la vía de escape de muchas personas. Además, con la introducción de Internet, también está siendo una forma más de socializar, conocer gente nueva y divertirse con los amigos.

- *E-mail de contacte: Xavier.Clemente@e-campus.uab.cat*
- *Menció realitzada: Computació*
- *Treball tutoritzat per: Alejandro Párraga (Computer Sciences Department)*

1.1 Motivación

Ninja runner es el resultado de una idea que surgió con la aparición de las primeras aplicaciones para móvil. Desde que llegó el primer ordenador a nuestra casa, más o menos por el año 2000, siempre he mostrado interés por la tecnología y concretamente por los videojuegos. Comúnmente, soñaba con la posibilidad de poder ser productor de aquello que me hacía disfrutar y desconectar de mis problemas cotidianos, pero la tecnología de entonces requería un conocimiento bastante avanzado y una inversión de tiempo que no me podía permitir.

Con el paso de los años, las herramientas y la calidad de las aplicaciones han mejorado hasta tal punto, que cualquier persona con paciencia puede ser capaz de crear un videojuego. Otro pilar básico del desarrollo de este proyecto ha sido los conocimientos adquiridos en el grado de ingenieria informática. Las bases de la programación, la inteligencia artificial y la gestión de proyectos, me han permitido agilizar el proceso de creación de una aplicación para Android de principio a fin. El detonador principal que impulsó la realización de este trabajo, fue la aparición

de Unity 4.0. Gracias a este entorno ha sido posible hacer física y tangible una simple idea.

1.2 Objetivos

Como en todo proyecto, uno de los primeros puntos a tratar es que se pretende conseguir y marcar hasta donde se quiere llegar. Los objetivos propuestos siguieron la forma **SMART** (e**S**pecificos, **M**edibles, **A**lcanzables, **R**ealistas, establecidos en un **T**iempo). Definirlos de esta manera facilitó en gran parte el trabajo y su seguimiento, debido a que cada uno contenía la información necesaria para ser claro y concreto.

Los objetivos propuestos en orden de realización fueron los siguientes:

- Diseñar el juego de forma global: Personajes, enemigos, escenario, ambiente e ítems.
- Búsqueda de información sobre el entorno (Unity2D) y creación de videojuegos.
- Generación de menús, escenarios y HUD (interfaz de usuario).
- Creación del personaje principal e interacción con el entorno: correr y saltar.
- Movimiento de la cámara y del escenario.
- Animación y generación de obstáculos aleatorios.
- Sistema de puntuación, ítems (monedas, frutas, munición) y disparo del personaje.
- Implementación de enemigos básicos y su inteligencia artificial.
- Sistema de recompensas para bonificar el progreso del jugador.
- Diseño, creación e inteligencia artificial del enemigo final.

2 Estado del arte

La historia de la creación de los videojuegos se sitúa a la par de las primeras computadoras en 1945 durante la Segunda Guerra Mundial. Inicialmente, los juegos eran simuladores de guerra para los pilotos o máquinas con inteligencia artificial para jugar al ajedrez, pero poco a poco, con la invención de los microprocesadores, estos fueron evolucionando y derivaron a géneros muy diversos.

Ninja runner pertenece al género de plataformas en 2D. Este tipo de juegos se remontan sobre los años 80 con el nacimiento de las máquinas arcade. Los títulos más conocidos de la época fueron Donkey Kong, Mario Bros (figura 1) y Sonic the Hedgehog. La mecánica principal consiste en un personaje que camina, corre, salta o escala por un escenario. La cámara suele usar vistas horizontales de izquierda a derecha, de manera que el protagonista tiene que avanzar consiguiendo objetos, superando enemigos y esquivando acantilados para poder completar el juego. Pese a la antigüedad de dicho género, sigue siendo uno de los principales en la actualidad.

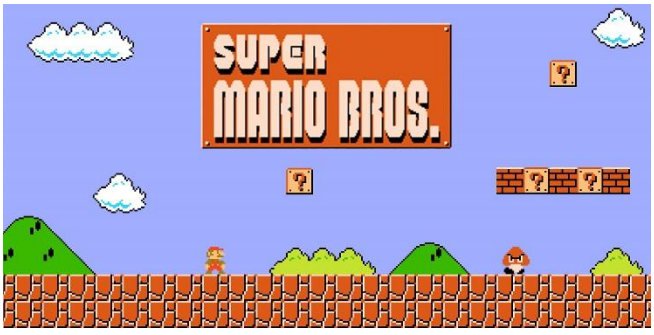

Figura 1. Juego de plataformas 2D: Mario Bros

A medida que la tecnología iba siendo cada vez más sofisticada, los videojuegos se iban adaptando a diferentes plataformas, dando lugar a las primeras videoconsolas portátiles. Este concepto se aprovechó con la invención del Smarthphone: ahora los móviles aparte de utilizarse para hacer llamadas, también sirven como una herramienta lúdica. El título primigenio por excelencia de este suceso fue Snake, creado por Nokia a finales de los años 90. Es un juego sencillo que consiste en una serpiente que tiene que capturar círculos para hacerse más larga y a su vez, el jugador debe evitar impactar contra sí mismo.

Actualmente, el mercado de los videojuegos para móviles es uno de los más importantes para la industria, superando en ingresos a plataformas famosas como Playstation de Sony o Xbox de Microsoft. Para hacernos una idea del dinero generado, podemos fijarnos en los títulos más populares como Clash of Clans, que obtuvo unas ganancias de 1345 millones de dólares en 2015, precedido por Candy Crush Saga con 682 millones o el más actual "boom" con el lanzamiento de Pokemon Go en 2016 que hizo ingresar 950 millones de dólares a la empresa Niantic. Para los más previsores, era de esperar tal popularidad, debido a que hoy en día casi todo el mundo de cualquier edad posee un smartphone a diferencia las videoconsolas que tienen un target mucho más reducido.

Por último, recalcar la mejora sustancial en estos últimos años: hemos pasado de tener juegos sencillos en blanco y negro con una mecánica muy básica a juegos en color, en 3D, realidad virtual con controles realmente sofisticados como manejo táctil o por sensor de movimiento. Es evidente la gran inversión que se ha realizado en este sector desde sus inicios y que, basándonos en la estadística, está en crecimiento exponencial.

3 METODOLOGÍA Y DESARROLLO

3.1 Metodología Ágile

La metodología seguida para la elaboración del TFG ha sido del tipo agile. A lo largo del grado hemos visto que una metodología agile es muy útil para proyectos cambiantes. Nos interesa este tipo de desarrollo porque al tratarse de un videojuego, requiere estar en constante cambio para adaptarse a los imprevistos y a las demandas de los usuarios.

Los sprints se hicieron cada dos semanas, de manera que Alejandro Párraga, el tutor del TFG, actuaba como scrum master, encargado de guiar el proyecto y marcar las fechas en las cuales deberían estar realizadas las actividades fijadas. Cada sprint estaba definido por una serie de tareas ordenadas por prioridad, donde se establecía un tiempo estimado y una descripción de cada una.

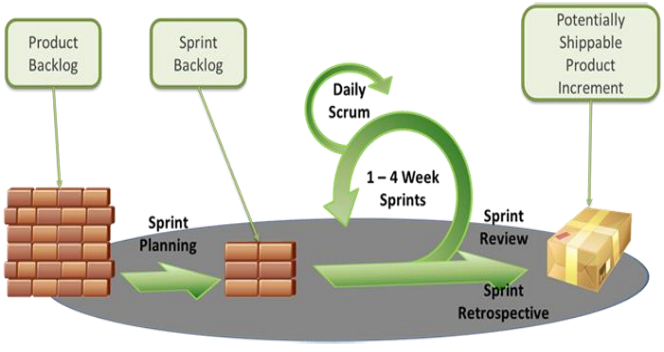

Figura 2. Metodología Agile

La planificación de las tareas se realizaba al principio de cada sprint. Los diferentes pasos que se siguieron fueron los siguientes:

- Identificación de las diferentes tareas a realizar y estructura del proyecto.
- Ver las limitaciones: plazo de entrega, programario disponible, viabilidad.
- Establecer una lista de recursos necesarios.
- Asignar una fecha y estimar una duración a cada tarea.
- Evaluar el trabajo realizado al final de cada sprint.

Para poder tener un feedback con el usuario de manera gradual, se moduló el software en versiones, de forma que cuando se realizaba una actualización de la aplicación, se le dejaba probar a una serie de testers para recibir una opinión. Dicha opinión nos sirvió para poder reajustar el proyecto en función de la demanda de la mayoría de jugadores, siempre que estuviera dentro de nuestras posibilidades.

3.2 Herramienta de desarrollo: Unity 4.0

Como hemos mencionado en puntos anteriores, la herramienta principal utilizada ha sido Unity 4.0 (figura 3). Este motor gráfico de videojuegos multiplataforma funciona mediante programación en C, C# [5] o C++. Se puede utilizar tanto en proyectos 2D como en 3D. La primera versión de Unity [1] funcional se presentó en Septiembre del 2010 y se han ido añadiendo mejoras hasta la actualidad.

Las ventajas de utilizar este editor frente a otros son evidentes. Cuenta con un editor que permite agrupar las escenas en un solo espacio de trabajo, además de ser intuitivo y fiable. Se puede desarrollar un videojuego de calidad sin necesidad de tener un equipo de diseño especializado y además es adaptable a todas las resoluciones. Posibilita el lanzamiento en numerosas plataformas sin realizar ninguna tarea de implementación.

Se divide en 5 vistas principales:

Inspector: Muestra y define las propiedades de los elementos del proyecto. Se pueden modificar los valores de forma rápida, cambia texturas arrastrando ficheros desde el Explorador, añade scripts, guarda prefabs [7].

Explorador: Lista todos los elementos (o activos) de los proyectos. Esta herramienta permite ordenar de forma intuitiva el proyecto. En esta vista se encuentran tus imágenes, escenas, scripts, audios, prefabs, texturas, atlas y todos los elementos.

Jerarquía: Lista en orden jerárquico de los elementos de la escena

Escena: Diseño y maqueta de tu juego completo o una pantalla o sección de éste. Cada escena representa un nivel o sección diferente del juego (portada, nivel 1, nivel 2, login…). Simplemente permite arrastrar tus activos desde el Explorador y edita sus variables desde el Inspector.

Juego: Es una vista WYSIWYG (What You See Is What You Get). Permite visualizar el juego a distintas resoluciones.

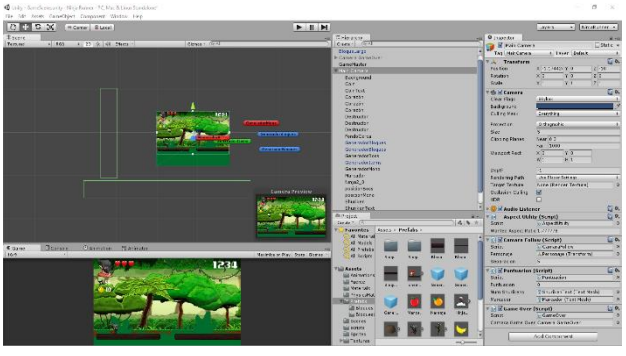

Figura 3. Editor Unity 4.0

3.3 Arte

El aspecto visual de un juego es una de las partes más importantes para el producto a la hora de atraer al usuario. Debido a la falta de habilidad artística y con programas de edición, se decidió utilizar el RPGMaker (figura 4) para diseñar los sprites del personaje y los enemigos. También se utilizaron sprites gratuitos producidos por otros usuarios de Unity como por ejemplo el fondo de la escena principal, las plataformas, los ítems o la interfaz de usuario (corazones, puntación…).

Figura 4. Creador RPGMaker

3.4 Enemigos e Inteligencia Artificial

3.4.1 Planta carnívora

A lo largo del recorrido aparecerá de manera aleatoria y bastante seguida este tipo de enemigo. Es el enemigo más sencillo de juego y creado en forma de obstáculo. La Inteligencia artificial implementada en este objeto es un detector de proximidad, la planta se mantiene estática hasta que el personaje principal se encuentre en un radio de unos 5 metros, entonces esta se activará y se elevará como se muestra en la figura 5. Para añadirle un poco de dificultad, a diferencia de otros enemigos, esta planta no se podrá destruir con un simple shuriken, para evitar perder una vida, el jugador tendrá que esquivarla saltando.

Figura 5. Enemigo 1: Planta Carnívora

3.4.2 Mono enfurecido

Como la planta carnívora, este enemigo también aparecerá de forma aleatoria a lo largo del recorrido, pero de forma menos frecuente y en la misma posición, ya que se trata de un enemigo estático y más complicado. Como podemos observar en la imagen de la figura 6, el mono está situado encima de una plataforma y nos irá lanzando rocas. Dichas rocas tratarán de impactarnos, cada impacto significará una vida menos para el jugador. La inteligencia artificial es bastante más sofisticada que la de la planta. Hay tres tipos de IA añadidas a este enemigo, todo dependerá del número de veces que se ha generado, controlado por una variable.

Durante la primera generación, este enemigo solo se limitará a lanzar piedras a la misma dirección, de manera que serán más previsibles y fáciles de esquivar para el jugador. En la segunda generación, se le añadirá un componente más de dificultad: el mono enfurecido alternará entre el primer tipo de lanzamiento y uno que apunta directamente a la posición del personaje. De esta manera, será mucho más difícil de intuir, ya que el usuario no sabrá qué tipo de lanzamiento utilizará. La tercera y última generación será la más complicada: nuestro enemigo esta vez alternará entre el disparo dirigido a la posición del personaje y un lanzamiento completamente aleatorio. Para este último lanzamiento, se han utilizado dos valores float random, uno entre 1-3 para la gravedad y otro entre 1-10 para la potencia. Modificar la gravedad comporta que la piedra caiga mucho más deprisa y el rebote contra una plataforma será menor. Por lo contrario, modificar la velocidad, implicará una disminución de tiempo en el que tarda la piedra en avanzar hacia la dirección del personaje.

Figura 6. Enemigo 2: Mono enfurecido

A diferencia de la planta, el mono enfurecido se podrá eliminar. Cuenta con una barra de vida compuesta por tres rectángulos rojos. Esta barra se puede reducir a base de shurikens lanzados de nuestro personaje principal. Si se consigue hacer impactar tres shurikens contra el mono, éste se destruirá y nos otorgará una puntuación extra.

3.4.3 Boss: Ninja asesino

El último tipo de enemigo y el más importante es el boss final, el ninja asesino. Su aparición se realiza tras eliminar los tres tipos de mono mencionados anteriormente y un lapso de tiempo entre 20 y 30 segundos. La posición inicial es estática en la parte derecha del escenario y está seguido por la cámara principal, como podemos observar en la figura 7. El movimiento básico es automático, encima de una nube el ninja va moviéndose de arriba a abajo para que sea un poco más complicado de impactar. Como el mono, el ninja enemigo se podrá eliminar con shurikens, con la diferencia que éste en vez de tres rectángulos, cuenta con una interfaz muy parecida a la del personaje principal, con una cara representativa y tres corazones en la parte superior de la pantalla.

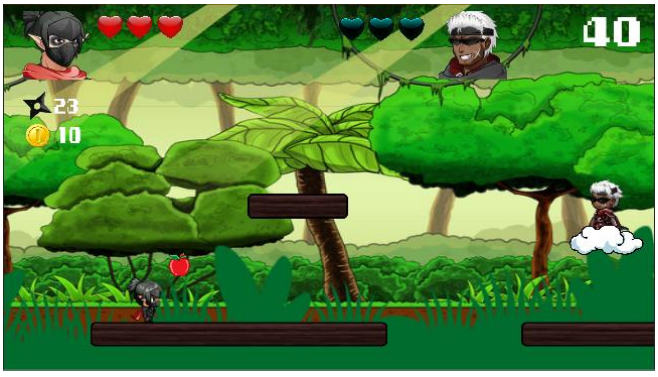

Figura 7. Enemigo 3: Ninja asesino

Por lo que hace a la inteligencia artificial implementada, este enemigo cuenta con la más compleja:

Lanzamiento de shuriken predictivo: Durante todo el recorrido del jugador hasta llegar al boss se hará un cálculo del número de saltos, la posición y el tiempo de media entre saltos para obtener una posición aproximada del jugador en un intervalo de tiempo determinado. Gracias a este cálculo, el jefe lanzará un shuriken cada X tiempo a la posición resultante. Esta implementación da una sensación de aprendizaje, es decir, el enemigo atacará de una manera u otra en función de la forma de jugar de cada jugador. Por ejemplo, a un jugador que utiliza mucho el doble salto, el lanzamiento le será mucho más elevado que un jugador que permanezca más tiempo en las plataformas de abajo.

Detector de shurikens: El ninja asesino cuenta con un detector situado a unos 5-6 metros en frente del personaje que le avisarán de cuando un proyectil está en camino. Cuando le llega la notificación de que va a ser impactado, hará un desplazamiento hacia arriba o hacia abajo, dependiendo de la dirección de movimiento automático en ese instante, para esquivarlo. El punto débil de este detector es que no es capaz de detectar dos lanzamientos en diferentes puntos, por lo que el jugador tendrá que tratar de predecir el movimiento, no servirá con un lanzamiento normal.

Detector de plataformas y bomba de humo: A lo largo del enfrentamiento, existe una probabilidad con un tiempo de recarga, de que lance una bomba de humo a una plataforma, de manera de que si el humo que deja la bomba impacta contra el personaje, éste quedará cegado. La ceguera se basa en que la visión se reduce, oscureciendo la pantalla donde sólo se podrá ver un circulo que rodea al ninja principal. El detector de plataformas indicará al ninja asesino donde puede depositar la trampa para que no la lance al vacío.

Reflector de Shurikens: Para evitar que el jugador abuse de un lanzamiento muy seguido de shurikens, este último componente contará el número de shurikens lanzados en un intervalo de 3 segundos. Si el detector cuenta más de tres, se activará una barrera reflectora que rebotará las siguientes estrellas ninjas, devolviéndolas a más velocidad durante los próximos 10 segundos.

3.5 Escenas

3.5.1 Escenario principal

La escena donde se desenvuelve la acción principal, está compuesta por un scroll parallax y los elementos producidos por los generadores. El funcionamiento del scroll consiste en dos fondos que se mueven a distinta velocidad, uno trasero y uno delantero. Jugando con las velocidades de los dos fondos se consigue una sensación de movimiento y un poco de profundidad. De esta manera, pese a ser un juego en 2D se crea un efecto visual del personaje corriendo por unas plataformas delanteras, mientras se va moviendo el fondo en la parte trasera.

3.5.2 Pantalla de inicio:

Figura 8. Pantalla inicial

Como la mayoría de juegos desde su creación, se ha añadido una pantalla de inicio. En la figura 8, podemos observar que la interfaz es bastante básica, está programada de la misma forma que el escenario principal, con la diferencia que se han eliminado todos los elementos del escenario y el personaje. En contrapartida, se ha agregado el título del

juego, el botón de jugar y la información de mi contacto. Haciéndolo bastante intuitivo, si se presiona el botón de jugar la escena del juego cambiará para dar paso a la posición inicial del personaje quieto a la espera de pulsar alguna parte de la pantalla que lo inicie.

3.5.3 Pantalla de GameOver

Figura 9. Pantalla Game Over

Cuando el jugador se cae de las plataformas o se le reduce el número de vidas a 0, el juego se acaba y aparece la imagen de la figura 9. Siguiendo la línea de la simplicidad, ésta nos muestra la puntuación realizada en la partida y la puntuación máxima conseguida en todos los intentos. Para no perder la puntuación máxima si cerramos la aplicación, se generará un archivo donde se anotará el número de puntos y este archivo se leerá al iniciar la aplicación. Por último, como en la pantalla inicial, cuenta con un botón de "jugar" que al ser presionado se reinicia el juego con todos los valores iniciales.

3.6 Items

3.6.1 Frutas

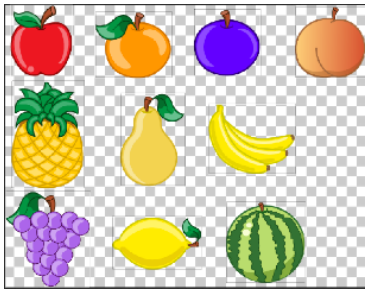

Figura 10. Item 1: Frutas en el editor de sprites

En la figura 10 podemos observar el primer tipo de ítem básico que nos encontraremos a lo largo del recorrido, las frutas. La única funcionalidad de estos ítems es otorgar puntuación extra al jugador. Cada fruta da una puntación distinta: naranja – 1pt, manzana – 5 pts, plátano – 10pts y sandía – 25pts.

3.6.2 Shuriken

Con el fin de poder hacer el ataque principal del ninja, se ha introducido el ítem de shurikens. En la interfaz del usuario se indica la cantidad de shurikens disponibles para lanzar. A lo largo del nivel, aparecerá este objeto de manera aleatoria encima de una plataforma para poder recargar la munición si se obtiene. Un punto importante de la mecánica de disparar, es que se ha implementado un algoritmo de aparición de los shuriken en función si tenemos mucha o poca munición: Si tenemos poca, el ratio de objetos shuriken aumentará por dos.

3.6.3 Monedas

Este objeto, a diferencia de las frutas, en vez de otorgarnos puntos nos dará monedas que se almacenarán en un "monedero". Haciendo una similitud con las monedas reales, funcionarán de forma parecida: En una posterior actualización se introducirá una tienda para poder comprar bonificaciones, trajes y poderes para que se apliquen al personaje principal dentro del juego, como por ejemplo, una vida extra, mayor número de munición, invulnerabilidad temporal, etc.

3.7 Generadores y destructores

Como hemos mencionado anteriormente, los elementos del escenario se generan de forma aleatoria. Si nos fijamos en la Figura 11, podemos observar tres tipos de generadores distinguidos por colores: Un generador de ítems (verde), dos generadores de bloques (azules) y dos generadores de enemigos, uno para el mono y otro para el boss (rojos). El de ítems generará un ítem cada X tiempo y el de bloques irá alternando bloques cortos, medianos y largos aleatoriamente durante un intervalo de tiempo concreto.

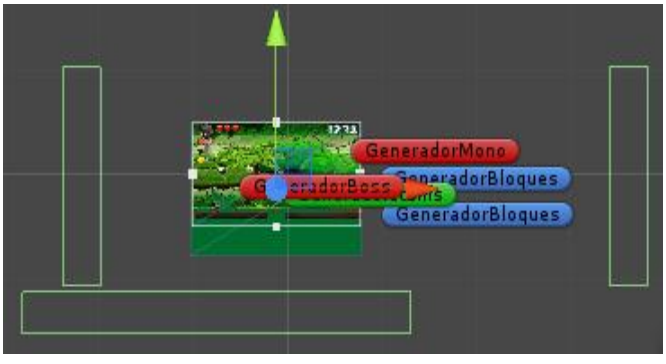

Figura 11. Generadores y destructores en la escena

Para no generar una sobrecarga de elementos en el escenario, se han creado tres destructores (los rectángulos verdes de la imagen) que avanzarán con la cámara, dos verticales y uno horizontal:

Los verticales, colocados a la izquierda y a la derecha del escenario principal, el de la izquierda para que a medida que avance la cámara y se vayan quedando elementos atrás, cuando estos impacten contra el destructor se eliminarán. El de la derecha es el encargado de eliminar los shurikens lanzados por el jugador.

El horizontal servirá para eliminar cualquier elemento que caiga al vacío, desde las piedras que lanza el mono enfurecido hasta el personaje principal. En este último caso, al entrar en contacto el personaje con el destructor, el jugador quedará eliminado y se mostrará la pantalla de Game Over.

3.7 Controles

Como hemos mencionado en párrafos anteriores, Ninja Runner es un juego que pertenece al género de plataformas 2D, concretamente del tipo infinity runner [3]. Como en cualquier juego de esta temática, los controles son bastante básicos: El personaje corre automáticamente, se puede saltar, hacer un doble salto y disparar.

Una de las características esenciales de la aplicación es su sencillez, por este motivo se decidió hacer los controles muy intuitivos. Al principio se pensó hacer un joystick, pero debido a la dificultad y viendo que realmente no era necesario, se optó por la opción de un canvas que dividía la pantalla del móvil en dos. La mecánica consistía en implementar un detector en la parte derecha y otro en la izquierda de la pantalla, de manera que si se detecta una presión en uno de los dos lados, el personaje hace una acción distinta. La parte izquierda se utiliza para saltar (si se presiona dos veces se puede realizar un doble salto) y la parte derecha para lanzar un shuriken. Se ha programado de manera que se puedan realizar dos acciones contiguas pero no a la vez: no se puede saltar y lanzar un shuriken a la vez, es decir, pulsar la zona izquierda y derecha al mismo tiempo, pero si se puede saltar y después lanzar un shuriken.

3.8 Animación

Uno de los componentes más importantes a la hora de darle dinamismo al juego es la animación. Cada animación está compuesta por un script que va cambiando los valores del sprite seleccionado (color, posición, ángulo, sprite render…). De esta manera, se ha podido animar el personaje principal, los enemigos y algunas de sus acciones. Por ejemplo, para hacer la animación de correr del protagonista (figura 12), se ha realizado un intercambio del sprite estático en cada frame por uno que mueve las piernas y así se consigue la sensación de que el objeto se esté moviendo.

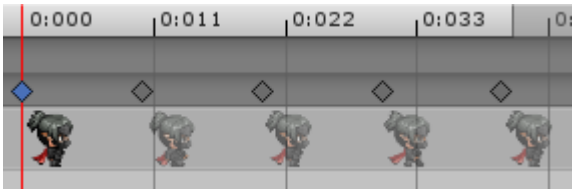

Figura 12. Secuencia de la animación de correr

Para poder realizar las transiciones entre animaciones [2] se ha utilizado la herramienta animator de Unity. Como podemos observar en la figura 13, cada animación es un estado y cada flecha es una transición. La transición se realiza si se cumple una condición, por ejemplo, si el personaje está corriendo y el jugador hace la acción de saltar, hay un booleano que controla dicha condición que es comprobar si el personaje está saltando, si su valor es true, se realizará la transición y en consecuencia, se cambiará la animación. La animación que se esté realizando en cada momento se pondrá de color naranja en el animator.

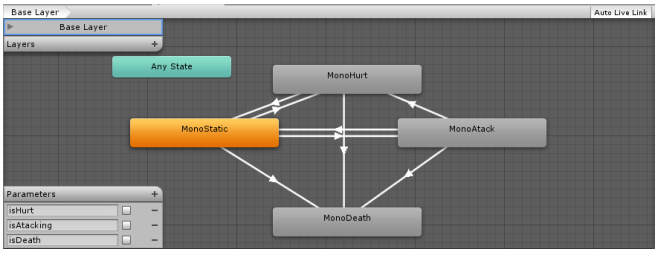

Figura 13. Árbol de animaciones

3.9 Colisiones

Gracias al gestor de colisiones ha sido posible realizar todo el sistema de vidas, eliminación de los enemigos y obtención de objetos, entre otras funcionalidades. A cada objeto que queríamos que interactuara con el entorno o con otros objetos, se le añadía un componente Collider2D. Como podemos observar en la figura 14, está representado mediante un círculo o cuadrado de color verde. Una colisión se produce en cuanto dos componentes Collider2D se cortan entre sí. Los objetos a los que se le añadió este componente fueron los siguientes: Personaje principal, ítems, enemigos, destructores, plataformas y los objetos lanzados por el personaje o los enemigos (shurikens, piedras).

Hay dos tipos de colisiones en Unity, las colisiones normales y las trigger. La diferencia entre ambas, es que en las colisiones trigger no tienen en cuenta la masa del objeto con la que impactan, es decir, si un objeto trigger impacta a otro, lo atravesará.

Como veremos a continuación, todas las interacciones con plataformas contienen un tipo de colisión normal. Esto es debido a que una plataforma tiene que ser solida, si fuera trigger sería atravesada, el personaje, las plantas y el mono se caerían al vacío. Además, gracias a este tipo de colisión, las piedras que lanza el mono enfurecido rebotan con la plataforma y hacen menos predecible el disparo.

Interacciones con colisión trigger:
- Personaje -> Ítem
- Shuriken -> Enemigo
- Piedra -> Personaje
- Planta -> Personaje
- Cualquier objeto -> Destructor

Interacciones con colisión normal:
- Personaje -> Plataforma
- Piedra -> Plataforma
- Mono -> Plataforma
- Planta -> Plataforma

Figura 14. Colliders2D de algunos objetos

Para terminar, otra funcionalidad del sistema de colisiones es la parte de "Physics2DSettings" del proyecto. En la figura 15 vemos como podemos regular que objetos queremos que colisionen. Esta parte es muy importante porque sin este control, se realizarían colisiones no deseadas, como por ejemplo, colisiones entre shuriken – ítem o piedra – planta, dando como resultado la eliminación del objeto impactado. Como es lógico, no queremos que los shuriken lanzados por el personaje principal destruyan las frutas o las monedas, o que las piedras lanzadas por el mono destruyan las plantas, entre otros.

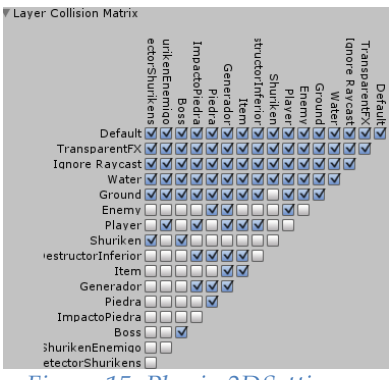

Figura 15. Physics2DSettings

3.10 Audio: música y efectos de sonido

Para finalizar con los componentes de Ninja Runner, el último añadido fue la música y los efectos de sonido. El proyecto cuenta con 3 tipos de música ambiental en bucle para cuando el personaje está corriendo sin ningún enemigo, 3 tipos de música para cuando aparecen los monos y por último, una canción para cuando aparece el enemigo final. Como efectos de sonido, se le ha añadido al ninja protagonista sonidos específicos cuando salta, lanza un shuriken o es impactado por una piedra o un ataque del ninja asesino.

5 RESULTADOS

En esta sección se describirá el resultado final del proyecto, así como todas las dificultades encontradas a lo largo de su elaboración, los cambios realizados y el análisis realizado por los testers de la aplicación.

5.1 Dificultades

La primera dificultad encontrada fue el manejo del entorno Unity 4.0. Como hemos descrito anteriormente, esta herramienta nos ha permitido realizar un videojuego de principio a fin sin necesidad de tener un equipo de gente. Pese a contar con un editor intuitivo, poder ser capaces de extraer todo el rendimiento no ha sido tarea sencilla. El primer paso fue familiarizarse con el entorno siguiendo tutoriales, documentación de la página oficial y consultando algunos libros de programación que se encuentran en la bibliografía. Una vez realizada la tarea de investigación del entorno, pasamos a la fase de prueba y error. Como en muchos proyectos de investigación, uno de los métodos de adquisición de conocimiento es realizar acciones y ver el resultado obtenido. Gran parte de las funcionalidades de la aplicación surgieron a raíz de varios intentos hasta dar con la solución.

La segunda dificultad que se nos planteó fue la optimización de los FPS (Frames por segundo) del videojuego. Debido a que las imágenes no estaban bien optimizadas, conforme se iban añadiendo más elementos, el número de FPS se reducía considerablemente, dando como resultado una ralentización notable en la jugabilidad. Para solucionar este problema se reescalaron muchas imágenes utilizadas (fondos, sprites de personaje, ítems y enemigos) para que no hubiera sobrecarga gráfica. Finalmente, después de hacer estas modificaciones, reduciendo el número de bucles y llamadas a condiciones, se consiguió optimizar el juego hasta el punto de que no hubiera molestia a la hora de jugar.

Por último, remarcar los bugs más comunes tras programar algunos scripts como por ejemplo problemas con el sistema de vidas. El personaje al ser impactado por una piedra del mono, en vez de restarse una vida, el ninja era desplazado hasta un barranco. Para solucionarlo se hizo un

objeto padre que contuviera dos hijos, cada uno con un Collider2D diferente. Un collider era normal, para que se aplicaran las físicas de la piedra y rebotara contra las plataformas y el otro collider era del tipo trigger, para que al impactar con el personaje no se apliquen las físicas, lo atraviese y le resta una vida. Otro error que ralentizó bastante el desarrollo, fue la animación de crecimiento de la planta carnívora, cuando se producía dicha animación, esta en vez de hacerse en la posición donde se encontraba la planta, se realizaba en la posición de la primera planta. Para enmendar este bug, se decidió aplicar la animación sobre el objeto padre, que era donde se generaba la planta, en vez de aplicarlo sobre la posición relativa donde se encontraba el sprite.

5.2 Cambios realizados

Al tratarse de un desarrollo de software ágil y que el proyecto trataba de ser una aplicación bajo la demanda de unos usuarios, se han realizado algunos cambios respecto a la planificación incial. Para mantener un feedback constante con los futuros jugadores, a cada modificación notable realizada sobre el producto, se lanzaba una versión nueva de la aplicación y se le dejaba a probar a un número entre 15 y 30 testers.
Las modificaciones realizadas fueron las siguientes:

- Aumentado el número de vidas a 6 debido a la dificultad del juego.
- Ajuste en la velocidad de lanzamiento de piedras del mono enfurecido.
- Disminución en el ratio de aparición de las plantas carnívoras.
- Aumento del ratio aparición de munición cuando el jugador está en mínimos.
- Disminución de la separación entre plataformas inferiores.
- Ajuste en el movimiento de esquivar del ninja asesino.
- Ajuste en la visibilidad producida por la bomba cegadora del ninja asesino.
- Cambio en el fondo del juego para mejorar su aspecto visual.
- Optimización de los FPS del videojuego para móviles menos potentes.
- Eliminada la idea de la tienda por falta de tiempo.
- Eliminada la idea de aparición de columnas verticales por exceso de dificultad.

La valoración global de los testers ha sido que Ninja Runner es un juego entretenido, colorido y que destaca por su sencillez. Se considera un juego difícil a la vez de adictivo, debido que llegar al boss es un reto bastante complicado y engancha al usuario a seguir jugando hasta conseguirlo. Por lo general la valoración es bastante buena y muchos de ellos añaden que les gustaría que hubiera una tienda, más personajes, powerups y más tipos de enemigos una vez eliminado el boss.

5.3 Estado actual

Ninja Runner se encuentra en un estado de funcionalidad total, se puede realizar una partida de principio a fin sin ningún tipo de bug grave que bloquee la aplicación. Los resultados obtenidos han sido los esperados y han seguido casi todos los puntos de la planificación inicial que podemos encontrar en el Anexo A.

Haciendo una recopilación de todos los elementos mencionados en la parte de desarrollo y viendo la figura 16, vemos que el videojuego cuenta con lo siguiente: Tres escenas con scroll parallax (fondo dinámico), un personaje principal, animaciones, controles propios, un sistema de generación de plataformas y objetos aleatorios, tres tipos de enemigos con inteligencia artificial, un sistema de vidas, puntuación, monedas y colisiones, algoritmos que reaccionan a la forma de jugar del usuario como el disparo dirigido del boss o el ratio de aparición de la munición, efectos de sonido y música ambiental dependiendo de la situación.

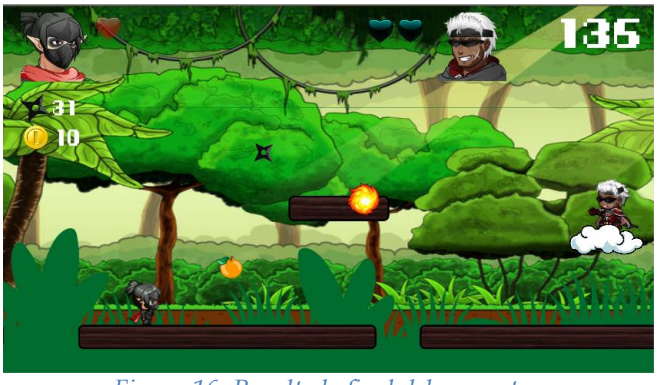

Figura 16. Resultado final del proyecto

6 Conclusiones

Como resultado del trabajo final de grado presentado, se puede concluir que es posible realizar un videojuego de principio a fin, con todos sus componentes gracias a la herramienta Unity. Remarcar los conocimientos adquiridos durante todo el proceso, no solo del entorno y de la programación en C#, sino también de como planificar un trabajo de 5 meses de duración, gestionar tareas y resolver los imprevistos de manera rápida y eficaz. Se podría decir que Ninja Runner es el conjunto de todos los conocimientos adquiridos durante el grado de Ingeniería Informática más un componente personal.

Para finalizar, cabe destacar que debido a la falta de tiempo, no se ha podido realizar muchas de las ideas iniciales y que por este motivo, no se incluyeron en la planificación. Entre ellas está la implementación de una tienda; actualmente las monedas no tienen ningún valor significativo porque no se pueden intercambiar por nada. En dicha tienda se incluirían trajes nuevos, poder aumentar el nú-

mero de vidas, aumentar el número de munición, obtención poderes adicionales para el ninja, invencibilidad temporal. Otras ideas son: que el escenario vaya cambiando conforme se obtiene más puntuación, que aparezcan más enemigos con distinta inteligencia artificial y crear un sistema de powerups. Es posible que en un futuro se mejore la aplicación y se adapte a otras plataformas como iOS con el fin de comercializarla, pero de momento queda en el aire debido a que más que un proyecto con un fin comercial, ha sido un trabajo de aprendizaje.

6.1 Agradecimientos

Quisiera mostrar mi agradecimiento a Alejandro Párraga, tutor de este proyecto, por guiarme, otorgarme la libertad de planificar las tareas de forma autónoma y por darme consejos útiles para realizar el trabajo de la mejor manera posible. También me gustaría darle las gracias a Cristian Castellano, por ser mi mentor desde que comencé los estudios, a Juan Díez y Arnau Boïgues, mis dos grandes amigos y compañeros de universidad, a Elena Maes y a mis rayitos: Laia, Sandra y Carmina, por darme ese apoyo moral necesario en los momentos difíciles. Por último, agradecer el soporte familiar durante todo el desarrollo del TFG.

7 Bibliografía y referencias

[1] Unity2D, https://goo.gl/7NMQ5y, *última consulta: 13/2/2017*

[2] Unity Animation, https://goo.gl/5rhlKG, *última consulta: 24/3/2017*

[3] InifnityRunner Information, https://goo.gl/jYOonX, *última consulta: 15/4/2017*

[4] EndlessRunner for Unity, https://goo.gl/ZMQfYT, *ultima consulta: 24/4/2017*

[5] "C# Programming with the Public Beta", escrito por Burton Harvey, Simon Robinson, Julian Templeman y Karli Watson y publicado por Wrox Press en 2000. *última consulta: 3/5/2017*

[6] "Professional C#", escrito por Simon Robinson, Burt Harvey, Craig McQueen, Christian Nagel, Morgan Skinner, Jay Glynn, Karli Watson, Ollie Cornes, Jerod Moemeka y publicado por Wrox Press en 2001.
última consulta: 6/5/2017

[7] Sprite editor, https://goo.gl/pIf7IF, *última consulta: 4/3/2017*

[8] Inifnity Runner Scripts, https://goo.gl/tCAqda, *última consulta: 7/4/2017*

[9] David Llansó, Pedro Pablo Gómez-Martín, Marco Antonio GómezMartín y Pedro Antonio González-Calero. A Declarative Domain Model can serve as Design Document. Papers from the 2013 AIIDE Workshop on Artificial Intelligence in the Game Design Process (IDP 2013). P. 9-15. Boston, Massachusetts, Estados Unidos. Octubre 2013. ISBN 978-1-57735-635-6 *última consulta: 17/3/2017*

[10] Unity prefabs, https://goo.gl/ET3G0Q, *última consulta: 14/5/2017*

[11] UnityList, http://unitylist.com/, *última consulta: 19/5/2017*

APÉNDICE

A. Planificación por sprints

Sprint 1 (1/2/17 – 12/2/17):
- ✓ Búsqueda de información.
- ✓ Definición de objetivos.

Sprint 2 (13/2/17 – 26/2/17):
- ✓ Planificación de tareas.
- ✓ Informe inicial.

Sprint 3 (1/3/17 – 12/3/17):
- ✓ Diseño del personaje.
- ✓ Diseño del escenario.
- ✓ Diseño de ítems.

Sprint 4 (13/3/17 – 26/3/17):
- ✓ Movimiento/salto del personaje y movimiento del escenario.
- ✓ Generar obstáculos aleatorios.
- ✓ Crear sistema de vidas.

Sprint 5 (27/3/17 – 9/4/17):
- ✓ Implementar sistema de puntuación.
- ✓ Implementar movimiento de la cámara.
- ✓ Informe de seguimiento I.

Sprint 6 (10/4/17 – 23/4/17):
- ✓ Animación del personaje.
- ✓ Diseño de enemigos.

Sprint 7 (24/4/17 – 7/5/17):
- ✓ Implementación de la IA de los enemigos básicos.
- ✓ Crear sistema de recompensas.
- ✓ Informe de seguimiento II.

Sprint 8 (8/5/17 – 21/5/17):
- ✓ Implementar poderes del personaje.
- ✓ Crear pantalla de inicio y game over.

Sprint 9 (22/5/17 – 4/6/17):
- ✓ Ajuste en el sistema de vidas y colisiones.
- ✓ Algoritmo de aparición de munición.

Sprint 10 (5/6/17 – 18/6/17):
- ✓ Diseño del boss: Ninja asesino.
- ✓ Configuración de la inteligencia artificial del boss.

Sprint 11 (19/6/17 – 30/6/17):
- ✓ Correciones finales y hacer la portabilidad a Android.
- ✓ Redacción del informe final y el paper.
- ✓ Preparar la presentación.

B. Glosario

Arcade: Es el término genérico de las máquinas recreativas de videojuegos disponibles en lugares públicos de diversión, centros comerciales, restaurantes, bares o salones recreativos especializados.

Smartphone: Es la familia de teléfonos móviles que disponen de un hardware y un sistema operativo propio capaz de realizar tareas y funciones similares a las realizadas por un ordenador, añadiéndole al teléfono funcionalidades extra a la realización y recepción de llamadas.

Script: Conjunto de órdenes guardadas en un archivo de texto, generalmente muy ligero y que es ejecutado por lotes o línea a línea, en tiempo real por un intérprete.

Prefab: Es un tipo de objeto en Unity prediseñado y reutilizable.

Variable: En programación, una variable está formada por un espacio en el sistema de almacenaje (memoria principal de un ordenador) y un nombre simbólico (un identificador) que está asociado a dicho espacio.

Sprite: objetos gráficos asociados a los diferentes elementos del juego.

Valor float: es un tipo de datos que almacena 32 bits de punto flotante. Se utiliza para representar valores decimales.

Bug: error o defecto en el software o hardware que hace que un programa funcione incorrectamente.

Shuriken: es un tipo de arma oculta arrojadiza como un proyectil. Está hecha de metal y es de origen tradicional japonés

C. Sprites adicionales

Personaje Principal:

Ninja Asesino:

Mono Enfurecido:

Planta Carnívora: